**AI Application design decisions for implementation - Rhys Richmond**

**Finite state machine:**

For the finite state machine which I have decided to use for my Ai application I have decided to go with a state machine which uses three virtual functions which are the stages of each state these are enter, update and exit, the main reason why I decided to do it this way is that it allows me to separate each state into sections so that everything happens in a specific order, comparing this to the switch case equivalent this is much more flexible as “With a switch statement, if you want to add a new class then you need to modify everywhere where the class is switched on” (Alexander, 2013).

By using the three virtual functions it allowed me to make it so that when it enters a new state, it does any state setup such as updating the strings for the states and actions. The update function is where it does the main part of the state for example getting the route to a location through pathfinding. Then finally the exit function is used to make changes to any variables needed and then gets the next state through a check state function.

As for the states for state machine, I made it so that it cycles through three of them based on a priority and checks of variables, which allows for the agents to act somewhat safer than normal as for example I have prioritised taking cover over shooting, so they would rather take less damage then deal damage.

**Pathfinding:**

The type of pathfinding which I have implemented in my Ai application is A\*. The reason for using A\* over the other types such as Breadth first search and Dijkstra’s algorithm is that A\* allows for is that “A\* is a good choice for most pathfinding needs.”(Patel, 2016). This is shown with performance, “The best thing to do is to eliminate unnecessary locations in your graph” as reducing the number of locations allows the algorithm to run faster and with A\* it uses a sum of both the calculation of the distance from the start point of Dijkstra’s and Greedy first search’s estimate of the distance to the goal point which allows it to get the optimal path while exploring less.

That is what I need in my application as I am using a somewhat small grid being 10x10 but since I am using pathfinding in all of my update functions of each state being able to do the pathfinding efficiently allows everything to run a lot smoother without worrying about performance related issues.

Another thing is that since all of my pathfinding is to a single location rather than multiple at the same time, it makes sense to use A\* for my pathfinding algorithm as the algorithm which you would use for single location pathfinding is either greedy best search and A\* and using A\* would be better than using greedy first search as you can change the heuristic for different needs. The other reason why I decided on using A\* in this case over Greedy first search is that “Greedy best-first tree search is also incomplete even in a finite state space, much like depth-first search.

(Russell, Stuart, and Peter Norvig, 2016)” and “It is not optimal”(Russell, Stuart, and Peter Norvig, 2016).

**Steering behaviours:**

The steering behaviours which I have used for my Ai Application is seeking and fleeing, the reason for this is that once I found the optimal path to location using A\* I then needed my agent to follow the path and the way I believe would be best to do this to use a seek function as it allows the agent to move along the path in a smooth and reliable way as it will update the velocity as it progresses as well as the rotation to make sure it's always looking in the correct direction.

The reason I decided to use seek instead pursuit is because I wanted the agent to go to the previous location the enemy agent was rather then following directly as “pursuit is the process of following a target aiming to *catch it*”(Bevilacqua, F, 2012) which is not want as it would lead to the agents tightly following the same path and would lead to both agents getting shot a large amount.

As for flee the reason I used in my application is for the helicopter object which when the soldier goes down to 0 health, it seeks the agent to give them more health and then once its reached them it flees from that position to make it seem like its trying to stay safe to get away from the warzone. The main reason for using flee over evade is due to the flee steering behaviour starting once the seek had been reached so using evade here would work as the helicopter would “flee away of the future position of the player”(IBRAHIM, 2013) but since its ready there this wouldn’t work properly.

**Bibliography:**

**Websites:**

Alexander, P. (2013). *c++ switch vs. member function pointer vs. virtual inheritance*. [online] Stack Overflow. Available at: https://stackoverflow.com/questions/20740639/c-switch-vs-member-function-pointer-vs-virtual-inheritance [Accessed 7 Jan. 2020].

Patel, A. (2016). *Red Blob Games: Introduction to A\**. [online] Redblobgames.com. Available at: https://www.redblobgames.com/pathfinding/a-star/introduction.html [Accessed 7 Jan. 2020].

Bevilacqua, F. (2012). *Understanding Steering Behaviors: Pursuit and Evade*. [online] Game Development Envato Tuts+. Available at: https://gamedevelopment.tutsplus.com/tutorials/understanding-steering-behaviors-pursuit-and-evade--gamedev-2946 [Accessed 9 Jan. 2020].

IBRAHIM, A. (2013). *Steering Behaviors*. [online] AH's Blog. Available at: https://ahmedhanibrahim.wordpress.com/2013/08/06/steering-behaviors/ [Accessed 9 Jan. 2020].

**Books:**

Russell, S, & Norvig, P 2016, Artificial Intelligence: A Modern Approach, Global Edition, Pearson Education Limited, Harlow, United Kingdom. Available from: ProQuest Ebook Central. [9 January 2020].

**Assets used:**

OpenGameArt.org. (2016). *Free Desert Top-Down Tileset*. [online] Available at: https://opengameart.org/content/free-desert-top-down-tileset [Accessed 7 Jan. 2020].

OpenGameArt.org. (2016). *top down soldier*. [online] Available at: https://opengameart.org/content/top-down-soldier [Accessed 7 Jan. 2020].

OpenGameArt.org. (2020). *Ammo Pack*. [online] Available at: https://opengameart.org/content/ammo-pack [Accessed 7 Jan. 2020].

OpenGameArt.org. (2014). *heli animated*. [online] Available at: https://opengameart.org/content/heli-animated [Accessed 8 Jan. 2020].